Claims 1-20 are pending in the present application. By this response, claims 1, 4, 5, 6, 11, 14, 15, 16 and 20 are amended. Support for the amendment to claims 1, 11, and 20 can be found in the specification at least at page 5, line 16 through page 6, line 8. Support for the amendment to claims 4, 5, 14, and 15 can be found in the claims as originally written. Support for the amendment to claims 6 and 16 can be found in the specification at least at page 6, lines 4-8 and page 35, lines 1-4. Reconsideration of the claims is respectfully requested.

I.   **Double Patenting Rejection: Claims 5, 15, and 20**

The Examiner provisionally rejects claims 5, 15, and 20, asserting as follows:

> Claims 5, 15, 20 are provisionally rejected on the ground of nonstatutory obviousness type double patenting as being unpatentable over claims 9, 19, 2 1 of copending Application No. 10,777,743 (hereinafter '743). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following observations.
>
> Following are but a few examples as to how the certain claims from the instant invention and from the above copending application are conflicting.
>
> **As per instant claim 5,** '743 claim 9 also recites first call tree for first build and second call tree for second build, subtracting the call tree for the second build from the call tree for the first build; outputting the subtracted call tree, wherein for each that exists in both the call tree for the first build and the call tree for the second build, generating a node in the subtracted tree data structure by subtracting a base value of the node in the second tree from a base value of a corresponding node in the first call tree. Although '743 claim 9 does not recite subtracted tree structure identifies differences between the execution of the first build and the second build; the effect of generating a subtracted structure from '743 wherein nodes are call tree nodes, the difference between execution of the respective build for the first and the second tree would have been a obvious limitation.
>
> **As per instant claim 15,** this is computer medium version of instant claim 5, reciting the same limitations therein, while '743 computer claim 19 corresponds to '743 claim 9; hence '743 claim 19 is a obvious variation of instant claim 15, by virtue of the above analysis.
>
> **As per instant claim 20,** '743 claim 21 also recites apparatus for generating first call tree for first build and second call tree for second build, subtracting the call tree for the second build from the call tree for the first build; outputting the subtracted call tree. Although the wording of claim 2 1 is not identical to that of instant claim 20, the subject matter of '743 claim 21 however would be merely an obvious variant of that of instant claim 20.
>
> This is a <u>provisional</u> obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Office Action dated October 23, 2007, pp. 2-3.

Applicants believe that the submission of a terminal disclaimer may be premature given the fact that the Examiner has made no indication as to whether claims of the 10/777,743 application are allowable. If at a later time such indication is made, Applicants may then elect to provide a terminal disclaimer for overcoming this provisional double patenting rejection.

Additionally, the amendments to the claims may vitiate the need for the terminal disclaimer. Applicants request that the Examiner reconsider this rejection and allow the claims.

II.   Objection to the Specification

The Examiner objects to the Specification, stating:

> The disclosure is objected to because some Application identifier format pertaining the 'Related Applications' (Specifications, pg. 1, top) still bears Attorney Docket number, and should be updated so they be expressed in proper U.S. Patent Application number format. Correction is required. See MPEP § 608.01 (b).

In response, Applicants have amended the specification in the manner set forth above. Therefore, this objection is overcome.

III.   35 U.S.C. § 112, Second Paragraph

The Examiner has rejected claims 6 and 16 under 35 U.S.C. § 112, second paragraph, as indefinite for failing to particularly point out and distinctly claim the subject matter that Applicants regard as the invention. This rejection is respectfully traversed. The Examiner asserts:

> Claims 6, 16 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.
> Claim 6 recites 'instructions for creating a node in the subtracting call tree structure having a negative value corresponding to a base value of the node that exists in either of the first call tree...or second call tree...'; and one of ordinary skill in the art cannot make use of the above teaching.
> According to the specifications, the subtracted node will be set value 2 when there is no corresponding node in the first tree for a corresponding second node tree as the second tree is being incrementally created during the process of call tree traversal (Specs, pg. 6, 2nd para; pg. 35, 1st para). The phrase *'having a negative value corresponding to a base value of the node that exists in either of the first call tree ... or second call tree'* cannot be commensurate with the Specifications, therefore appears to be out of context or obscure in terms of proper and feasible details for implementing the invention. One would not be able to construct a *subtractive tree structure* based on the above incongruous step thus recited. The negative value is treated as a subtraction from a negative value, as in an addition; or a subtraction where the amount being subtracted from has lesser quantity that the subtracting amount (subtractor).
> Correction is required.

Office Action dated October 23, 2007, pp. 4-5.

In response, claims 6 and 16 have been amended in a manner consistent with portions of the specification as identified above. Therefore, the rejection of claims 6 and 16 under 35 U.S.C. § 112, second paragraph has been overcome.

IV.   35 U.S.C. § 103, Obviousness: Claims 1-20

The Examiner has rejected claims 1-20 under 35 U.S.C. § 103(a) as being unpatentable over *Levine, et al.*, Method and System for Compensating for Instrumentation Overhead in Trace Data by Computing Average Minimum Event Times, U.S. Patent No. 6,349,406 (February 19, 2002) (hereinafter "*Levine*"), in view of *Reissman et al.*, Method and System for Automatically Testing a Software Build, U.S. Patent Publication No. 2005/0071818 (March 31, 2005) (hereinafter "*Levine*"). This rejection is respectfully traversed.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103(a). *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The scope and content of the prior art are determined; differences between the prior art and the claims at issue are ascertained; and the level of ordinary skill in the pertinent art resolved. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). Against this background, the obviousness or non-obviousness of the subject matter is determined. *Id.* Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue. *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007). Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. *Id.* (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006)). In the case at hand, the Examiner has not stated a *prima facie* obviousness rejection because the combination of references does not teach or suggest all the features of the independent claims.

Amended claim 1 is as follows:

> 1.   A method, in a data processing system, for identifying differences between the execution of a first build of a computer program and a second build of a computer program, comprising:
>   obtaining a first call tree data structure corresponding to first trace data of an execution of the first build of the computer program;
>   obtaining a second call tree data structure corresponding to second trace data of an execution of the second build of the computer program;
>   copying the first call tree data structure to form a copied call tree data structure;

subtracting the second call tree data structure from the copied call tree
data structure to generate a subtracted call tree data structure; and
outputting the subtracted call tree data structure, wherein the subtracted
call tree data structure identifies differences between the execution of the first
build of the computer program and the execution of the second build of the
computer program.

IV.A.   The Combination of References, Considered as a Whole, Does Not Teach or Suggest all of the Features of Claim 1

With respect to amended claim 1, the Examiner has not stated *prima facie* obviousness rejection because the proposed combination of references, when considered as a whole, does not teach or suggest the features, "copying the first call tree data structure to form a copied call tree data structure" and "subtracting the second call tree data structure from the copied call tree data structure to generate a subtracted call tree data structure," recited by amended claim 1.

Levine does not teach or suggest the feature, "copying the first call tree data structure to form a copied call tree data structure," recited by claim 1. For example, *Levine* discloses:

A method and system for compensating for instrumentation overhead in trace
data by computing average minimum event times is provided. In order to profile
a program, the program is executed to generate trace records that are written to a
trace file. A set of trace event records are processed, and the trace events are
represented as one or more nodes in a tree data structure. One or more
performance statistics are stored at each node in the tree data structure, and a
performance statistic at each node is processed to determine an overhead
compensation value. The overhead compensation value is determined by
computing a local overhead value for each node in the tree data structure. The
total execution time of a routine corresponding to the event represented by the
node is retrieved, and the local overhead value is computed as the average of the
execution time over the number of calls to the routine and the number of calls
from the routine to other routines. The minimum of all of the local overhead
values is the maximum possible global value used as the overhead compensation
value. The overhead compensation value is then applied to the performance
statistic at each node.

Levine, Abstract.

In this passage, *Levine* discloses a method and system for compensating for instrumentation overhead in trace data by using trace records represented as nodes in a tree data structure. In addition, *Levine* provides that the overhead compensation value is determined by computing a local overhead value for each node in the tree data structure. Importantly, neither the abstract nor any other portion of *Levine* discloses copying a call tree data structure to form a copied call tree data structure, as recited in amended claim 1. Instead, *Levine* discloses only that calculations are performed with respect to trace events stored in nodes of a single data tree structure without copying a call tree data structure. Thus, *Levine* does not

teach or suggest the feature, "copying the first call tree data structure to form a copied call tree data structure," as recited in amended claim 1.

Furthermore, *Levine* does not teach or suggest the feature, "subtracting the second call tree data structure from the copied call tree data structure to generate a subtracted call tree data structure," as recited by amended claim 1. Notwithstanding the fact that *Levine* does not teach or suggest a copied call tree data structure as discussed above, *Levine* also does not disclose *a second call tree data structure*. The Examiner believes otherwise, citing to the following portion of *Levine* as teaching subtraction with respect to a second tree data structure:

FIG. 20A
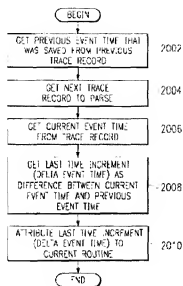
Levine, Figure 20A.

This figure of *Levine* discloses steps of a process for determining the manner in which elapsed time is attributed to various routines in an execution flow. In particular, Figure 20A discloses determining a delta event time for a current routine by calculating the difference between a current event time from a current trace record and a previous event time from a previous trace record. However, further analysis of Figure 20A in light of other portions of *Levine*, such as the Abstract, indicates that *Levine* discloses subtracting event times from different trace records stored in nodes of *a single tree data structure* rather than subtracting a *second call tree data structure from a copied call tree data structure* to generate a subtracted call tree data structure, as recited by amended claim 1.

For example, the abstract of *Levine*, reproduced above, discloses that "[a] set of trace event records are processed, and the trace events are represented as one or more nodes in *a tree data structure*. One or more performance statistics are stored at each node in the tree data structure, and a performance

statistic at each node is processed to determine an overhead compensation value." In other words, *Levine* discloses that the trace event records, which include the event times described in Figure 20A, are stored in nodes of a single data tree structure. Calculations are made using trace records of nodes in a single data tree structure rather than between a second data tree structure and a copied data tree structure, as recited by amended claim 1. Therefore, because *Levine* does not disclose either a copied call tree data structure or *a second call tree data structure*, *Levine* does not teach or suggest the feature, "subtracting the second call tree data structure from the copied call tree data structure to generate a subtracted call tree data structure," as recited by amended claim 1.

In addition, because *Reissman* does not cure the deficiencies of *Levine*, the proposed combination of *Levine* and *Reissman* also does not teach or suggest all the features of amended claim 1. Consequently, the Examiner does not to state a *prima facie* obviousness rejection as to amended claim 1.

For example, *Reissman* does not teach or suggest the feature, "copying the first call tree data structure to form a copied call tree data structure," recited by amended claim 1, because *Reissman* is entirely devoid of any reference to copying of call tree data structures. Instead, *Reissman* discloses only creating and comparing dictionaries that may take the form of tree data structures. Importantly, *Reissman* makes no reference to copying a first call tree to form a copied call tree during the creation and comparison of the dictionaries. For example, *Reissman* discloses:

> A system and method for improved testing of a software build is provided. The system and method automatically track differences between software builds by scanning the binaries of a software product to automatically discover its classes. The system and method then build a detailed dictionary that captures static and dynamic information of that build, including class dependencies. A comparison may be made with another build, so that the present invention may automatically execute selective tests on any types, and their dependencies, that have had a structural or behavioral modification since the last build. Testers may load a set of constructors for any specific types to further increase coverage of types (or classes) tested. The present invention may also provide detailed reports that may be used to drive future testing work and target specific areas of the code for additional testing. The system and method may further provide code generation from intermediate code to specified targets to aid in reproducing and fixing bugs.

Reissman, Abstract.

This Abstract of *Reissman* discloses dictionaries that may be generated based upon captured static and dynamic information of a software build. The dictionaries may then be compared. *Reissman* describes this comparison of the dictionaries in the following passage:

> FIG. 14 presents an illustration generally representing an example of a delta computed between two software builds. The differences discovered by a comparison between a new dictionary and an old dictionary may be recorded by listing the dictionary entry with the prefix <OLD> if deleted from the new dictionary and listing the dictionary entry with the prefix <NEW> if added to the

> new dictionary as shown in FIG. 14. For example, dictionary entry 1402 has the
> prefix <OLD> 1404 and indicates an entry that has been deleted from the new
> dictionary. Likewise, dictionary entry 1412 has the prefix <NEW> and indicates
> an entry that has been added to the new dictionary.

Reissman, p. 7, para. 66.

In this passage, *Reissman* describes the comparison of two dictionaries, each of which relate to a separate software build. In particular, *Reissman* describes the modification of entries of an existing dictionary to include a prefix identifying the dictionary from which the entry originated. Notably, nowhere does *Reissman* disclose that this comparison involves copying a first call tree to form a copied call tree, as recited by amended claim 1. Consequently, *Reissman* does not teach or suggest this missing "copying" feature.

In addition, because *Reissman* does not teach or suggest a copied call tree as discussed above, *Reissman* also does not teach or suggest the feature, "subtracting the second call tree data structure *from the copied call tree data structure* to generate a subtracted call tree data structure," recited by amended claim 1. Thus, neither *Levine* nor *Reissman*, singularly or in combination, teaches or suggests the "copying" and "subtracting" features recited by amended claim 1. Consequently, the Examiner has not stated a *prima facie* obviousness rejection as to claim 1.

Additionally, because independent claims 11 and 20 recite substantially the same features of independent claim 1, the proposed combination of *Levine* and *Reissman* also does not teach or suggest the features of claims 11 and 20 for the reasons set forth above. Furthermore, the Examiner has not stated a *prima facie* obviousness rejection as to claims 2-10 and 12-19 by virtue of their dependency from at least one of claims 1 and 11. Therefore, the rejection of 1-20 under 35 U.S.C. § 103(a) has been overcome.

IV.B.   The Examiner Failed To State a Proper Reason To Achieve the Legal Conclusion of Obviousness under the Standards of *KSR Int'l.*

Still further, the Examiner only provided a purported advantage to combine the references, but failed to connect that purported advantage to the *legal conclusion* of obviousness, are required by *KSR Int'l.* Under *KSR Int'l.*, the examiner must provide some rational underpinning to the *legal conclusion* of obviousness, not just state a purported advantage and then <u>assume</u> the legal conclusion of obviousness.

In the case at hand, the Examiner states that claim 1 would be obvious in view of the references because:

> executables intended for different environments (see Reissman) via improved
> builds can be addressed by profiling and analysis based on events as set forth by
> either Levine and Reissman, whereby based on such differences between trace
> profiling data, one build as well as testing techniques therefore can be readjusted
> for improvement over the previous build (see Reismann, BACKGROUND).

Office Action dated October 23, 2007, pp. 6-7.

Thus, the Examiner asserts that the claim would be obvious because *Levine* and *Reissman* could be combined to achieve the purported technical advantage. However, this type of reasoning does not comport with the requirements of *KSR Int'l.* because a rational underpinning for the legal conclusion of obviousness is not the same as an advantage. For example, one of ordinary skill would have to recognize the purported advantage, have a reason to implement the purported advantage, and also have no reason to avoid implementing the purported advantage in order to make the connection that one of ordinary skill would make the connection between the references in the first place, much less reach the *legal conclusion* of obviousness. Additional logic would be required to state a compelling case for the *legal conclusion* of obviousness of the claim at issue; simply reciting an "advantage" is not enough. Therefore, under the standards of *KSR Int'l.*, the Examiner failed to provide a rational underpinning to achieve the legal conclusion of obviousness. Hence, the Examiner failed to state a *prima facie* obviousness rejection against claim 1 or any other claim.

V. Conclusion

The subject application is patentable over the cited references and should now be in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: January 23, 2008

Respectfully submitted,

/Theodore D. Fay III/
Theodore D. Fay III
Reg. No. 48,504
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants